MaskRay

2013-08-27

约瑟夫问题的两个O(log n)解法

这个是学习编程时的一个耳熟能详的问题了：

`n`个人(编号为`0,1,...,n-1`)围成一个圈子，从`0`号开始依次报数，每数到第`m`个人，这个人就得自杀，之后从下个人开始继续报数，直到所有人都死亡为止。问最后一个死的人的编号(其实看到别人都死了之后最后剩下的人可以选择不自杀……)。

这个问题一般有两种问法：

- 给出自杀顺序。不少数据结构初学书都会以这个问题为习题考验读者对线性表的掌握。比较常见的解法是把所有存活的人组织成一个循环链表，这样做时间复杂度是`O(n*m)`的。另外可以使用order statistic tree(支持查询第k小的元素以及询问元素的排名)优化到`O(n log n)`。另外有篇1983年的论文*An O(n log m) Algorithm for the Josephus Problem*，但可惜我没找到下载链接。
- 求出最后一个人的编号。可以套用上一种问法的解法，但另外有更加高效的解法，下文展开讨论。

时间O(n)，空间O(1)

设`f(n)`为初始有`n`人时最后一个自杀的人的编号，那么有如下递推式：

```
1 f(n) = (f(n-1) + m) mod n
```

以`n=5, m=3`为例，一开始有这么5个人：

```
1 0 1 2 3 4
```

第一轮报数后`2`号死亡，圈子里剩下来的人的编号是这些：

```
1 3 4 0 1
```

这里把3号写在最前面了，因为轮到3开始报数。如果我们有办法知道n=4时的答案，即初始圈子为：

```
1 0 1 2 3
```

时的答案，那么可以把n=4的初始圈子的所有数x变换成(x+3) mod 5，得到：

```
1 3 4 0 1
```

这个恰好就是n=5时第一轮结束后的圈子状态，也就是说我们可以根据n=4的答案推出n=5时的答案。

手工演算一下就能发现n=z时的圈子第一轮结束后(即m-1号自杀后)的圈子状态，可以由n=z-1的初始圈子施以变换x -> (x+m) mod z得到。于是我们可以从n=1开始(此时的答案是0)，推出n=2的答案，再推出n=3，直到计算到所要求的n。

下面是C语言实现：

```
1 int f(int n, int m)
2 {
3   int s = 0;
4   for (int i = 2; i <= n; i++)
5     s = (s + m) % i;
6   return s;
7 }
```

时间O(log n)，空间O(log n)

换一个递推思路，考虑报数过程又回到第0个人时会发生什么。这时有floor(n/m)*m个人都已经报过数了，并且死了floor(n/m)人。之后一轮的报数会数过m个人，又会回到第0个人。

我们以n=8，m=3为例看看会发生什么。一开始：

```
1 0 1 2 3 4 5 6 7
```

floor(n/3)*3个人都报过数后：

```
1 0 1 x 3 4 x 6 7  (A)
```

即2号和5号死亡，接下来轮到6号开始报数。因为还剩下6个人，我们设法做一个变换把编号都转换到`0~5`：

```
1 2 3 x 4 5 x 0 1  (B)
2           ___
```

`(B)`的答案等于规模缩小后的情况`n'=6`时最后一个人的编号`s`。然后根据`s`算出圈子状态`(A)`的最后一个人的编号。

如果`s`在最后一个`x`后面(带下划线)，即`s < n%m`，那么`s2 = s-n%m+n`；否则`s2 = s-n%m+(s-n%m)/(m-1)`；

注意如果`n < m`，那么报数回到第一个人时还没死过人。因为子问题的规模没有减小，所以上述方法不适用。需要用之前时间复杂度`O(n)`的方法递推。下面是C语言实现：

```
1 int rec(int n, int m)
2 {
3   if (n == 1) return 0;
4   if (n < m) return (rec(n - 1, m) + m) % n;
5   int s = rec(n - n / m, m) - n % m;
6   return s < 0 ? s + n : s + s / (m-1);
7 }
```

`n`每次变为`n * (m-1) / m`，即以一个常数因子衰减到刚好小于`m`，然后换用线性复杂度的递推算法，总的时间复杂度为`O(m + log_{m/(m-1)}{n/m})`，如果把`m`看作常数的话就是`O(log n)`。程序中在子问题计算后还需要做一些处理，无法尾递归，所以空间复杂度也等于这个。

参考：

- Time Improvement on Josephus Problem (2002) by Fatih Gelgi
- http://stackoverflow.com/questions/4845260/josephus-for-large-n-facebook-hacker-cup

时间O(log n)，空间O(1)

三年前我还看到过一段更巧妙的代码，具体写法不可考了，当时盯着式子思考了好久呢。其等价的形式长这样：

```
1 int kth(int n, int m, int k)
2 {
3   if (m == 1) return n-1;
4   for (k = k*m+m-1; k >= n; k = k-n+(k-n)/(m-1));
```

```
5   return k;
6 }
```

这个函数解决了第二种问法的一个扩展问题，即第k个(从0数起)自杀的人的编号。如果取k = n-1那么就得到了最后一个自杀的人的编号。

这个算法的思想是第k*m+m-1次报数的人就是第k个自杀的人，追踪他之前每次报数的时刻来确定他的编号。考虑n=5, m=3, k=5，一共有15次报数(每报m次数死一个人，一共有k+1个人要死，所以需要(k+1)*m次报数)，每一次报数的人的编号如下：

```
1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
2 0 1 2 3 4 0 1 3 4 1  3  1  3  3  3
```

报到2、5、8、11、14的人自杀。

设第p次(从0数起)报数的人是y，令p = m*a+b (0 <= b < m)。经过前p次报数，一共死了floor(p/m) = a人，圈子里还剩下n-a人。如果y本次报数后没有自杀，那么他下次报数是在圈子里其他n-a人都报数之后，也就是第q = p+n-a = n+(m-1)*a+b次报数。这是后继的计算式，逆用即可计算前驱。

由y本次报数后还存活知b < m-1，故a = floor((q-n)/(m-1))，p = q-(n-a) = q-n+floor((q-n)/(m-1))。

我们要求第k个自杀的人，他是第k*m-1次报数后自杀的，用计算前驱的方式求出这个人之前一次报数是在什么时候，不断地重复这一过程，直到知道他是第k' (0 <= k' < n)次报数的人，那么k'就是这个人的编号。

algorithm, josephus problem

Comments

Popular

- 根据计划求解Rush (86)
- GTK+ tic-tac-toe（井字棋） (81)
- 用rss2email阅读feeds (80)
- 称球问题扩展——根据当前已知信息选择最优称量方案 (72)
- 给xbindkeys添加key seque... (71)
- genkernel (60)
- USACO JAN10 Gold (54)
- ZJU 1638. Greedy Island (51)
- Dancing Links+Algorit... (42)
- Awesome 3 debian menu (40)
- .emacs (38)
- FFT (38)
- XV Olimpiada Informat... (28)
- checker (13)

Tag Cloud

adc ai9 algorithm asc automaton awesome bctf binary bmc build system c c++ cgc chroot codinsanity coffee script compiler computer security contest csv ctf data structure debug defcon desktop docker elf emacs email emoji emscripten event expect feeds firmware floating point forensics game gcc gentoo graph drawing gtk hanoi haskell hpc inotify ipsec irc isc j javascript josephus problem jq kernel ld leetcode linux makefile math maze mirror ml mutt n-body network nginx nlp node.js noip notmuch npm ocaml offlineimap oi oj parallel parser generator perl presentation puzzle python qq regex regular expression reverse engineering review ruby ructfe scheme search security shell ssh stringology student festival puzzle suffix array suffix automaton summary suricata telegram telegramircd terminal traversal tree trendmicro udev unicode usb vim vpn vte wargame web analytics webqqircd website wechat wechatircd window manager xbindkeys xmonad yanshi

Blogroll

- BYVoid
- fqj1994
- ppwwyyxx